# Gentle Nearest Neighbors Boosting over Proper Scoring Rules

Richard Nock, Wafa Bel Haj Ali, Roberto D'Ambrosio, Franck Nielsen, Michel Barlaud

# Gentle Nearest Neighbors Boosting over Proper Scoring Rules

Richard Nock, Wafa Bel Haj Ali, Roberto D'Ambrosio, Frank Nielsen (*Senior Member, IEEE*), Michel Barlaud (*Fellow, IEEE*)

**Abstract**—Tailoring nearest neighbors algorithms to boosting is an important problem. Recent papers study an approach, UNN, which provably minimizes particular convex surrogates under weak assumptions. However, numerical issues make it necessary to experimentally tweak parts of the UNN algorithm, at the possible expense of the algorithm's convergence and performance. In this paper, we propose a lightweight alternative algorithm optimizing proper scoring rules from a very broad set, and establish formal convergence rates under the boosting framework that surprisingly compete with those known for UNN. It is an adaptive Newton-Raphson algorithm, which belongs to the same lineage as the popular Gentle Adaboost. To the best of our knowledge, no such boosting-compliant convergence rates were previously known for these algorithms. We provide experiments on a dozen domains, including the challenging Caltech and SUN computer vision databases. They display that GNNB significantly outperforms UNN, both in terms of convergence rate and quality of the solution obtained, and GNNB provides a simple and efficient contender to techniques that can be used on very large domains, like stochastic gradient descent — for which little is known to date. Experiments include a divide-and-conquer improvement of GNNB which exploits the link with proper scoring rules optimization.

## 1 Introduction

Iterative approaches to learn classifiers have been playing a major role in machine learning and statistical learning for at least forty years [1]. The most common high-level scheme consists in gradually combining from scratch classifiers obtained at each iteration, with the objective to minimize throughout iterations a convex differentiable risk called a *surrogate risk*, sometimes amended with a structural part based on data [2]. Unlike so-called greedy algorithms, that repeatedly perform fine-grained optimization steps [2], *boosting* algorithms rely on weak optimization stages much less demanding from the statistical and computational standpoints [3], [4], [5], [6], [7]. In fact, the boosting theory involves at each iteration weak classifiers slightly different from pure random, *but* requires that the final combination be probably as close as required from the optimum, within polynomial time.

Nearest neighbors (NN) rules are a non-trivial field of choice for boosting algorithms [4], [5], as examples ideally play weak classifiers. In this case, we treat the boosting problem in its simplest form: the accurate leveraging of examples that vote among nearest neighbors. In particular, we compute nearest neighbors in the ambient space of data, *i.e.* as described over their initial features. There have been other approaches to boost nearest neighbors by learning features with (Ada)boosting algorithms, prior to computing nearest neighbor rules on these new sets of features [8] (and references therein). No boosting results are known for these algorithms, and it is in fact not known whether they achieve convergence to the optimum of Adaboost's exponential risk. A previous approach in our line of works is algorithm UNN (for "Universal Nearest Neighbors"), which brings boosting guarantees for merely all strictly convex differentiable surrogates relevant to classification [9], [5], [6]. For a wide subset of surrogates, it yields simple and efficient estimators of posteriors [10].

There is, however, an analytical and computational bottleneck in UNN, as the leveraging coefficients are solutions to non-linear equations with no closed form expression in the general case. Boosting compliant approximations are possible, but in the context of NN rules, they are computationally far too expensive to be performed at *each* boosting iteration on large datasets. Computationally affordable coarse-grained approximations are also possible, that yield compelling experimental results, but it is not known if they always lie within the boosting regime [5].

In this paper, we propose a simple boosting compliant solution to this computational bottleneck. Our algorithm, GNNB for "Gentle Nearest Neighbors Boosting", performs adaptive Newton-Raphson steps to minimize any *balanced convex surrogate* [11] with guar-

*R. Nock (**contact author**) is with the Université Antilles-Guyane, CEREGMIA-UFR DSE, Campus de Schoelcher, B.P. 7209, Schoelcher 97275, France. E-mail: rnock@martinique.univ-ag.fr*
*W. Bel Haj Ali is with CNRS - U. Nice, France. E-mail: belhajal@i3s.unice.fr*
*R. D'Ambrosio is with University Campus Bio-Medico of Rome, Rome, Italy. E-mail: r.dambrosio@unicampus.it*
*F. Nielsen is with Sony Computer Science Laboratories, Inc., Tokyo, Japan. E-mail: nielsen@lix.polytechnique.fr*
*M. Barlaud is with Institut Universitaire de France and CNRS - U. Nice, France. E-mail: barlaud@i3s.unice.fr*

| | $\psi_\phi$ | $\phi$ | $\pi^*$ | $\delta_j(1/2)$ | weight update, $f : w_i \leftarrow f(w_i)$ |
|---|---|---|---|---|---|
| A | $(1-x)^2$ | $-x(1-x)$ | $\frac{1}{16}$ | $\frac{\eta(c,j)}{2n_j}$ | $w_i - 2\delta_{jc}y_{ic}y_{jc}$ |
| B | $\log_2(1+\exp(-x))$ | $x\ln x$ $+(1-x)\ln(1-x)$ | $\frac{\ln 2}{8}$ | $\frac{4\ln(2)\eta(c,j)}{n_j}$ | $\frac{w_i}{w_i \ln 2+(1-w_i \ln 2)\times\exp(\delta_{jc}y_{ic}y_{jc})}$ |
| C | $\log_2(1+2^{-x})$ | $x\log_2 x$ $+(1-x)\log_2(1-x)$ | | $\frac{4\eta(c,j)}{\ln(2)n_j}$ | $\frac{w_i}{w_i+(1-w_i)\times 2^{\delta_{jc}y_{ic}y_{jc}}}$ |
| D | $-x+\sqrt{1+x^2}$ | $-\sqrt{x(1-x)}$ | $\frac{1}{8}$ | $\frac{\eta(c,j)}{n_j}$ | $1-\frac{1-w_i+\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}{\sqrt{1+\delta_{jc}^2 w_i(2-w_i)+2(1-w_i)\sqrt{w_i(2-w_i)}\delta_{jc}y_{ic}y_{jc}}}$ |
| E | $\frac{1}{2}x(\mathrm{sign}(x)-1)$ | $-\min\{x,1-x\}$ | N/A | | |

TABLE 1
From left to right: examples of balanced convex losses $\psi_\phi$ (A, B, C, D; we let $\ln$ denote the base-$e$ logarithm, and $\log_z(x) \doteq \ln(x)/\ln(z)$); permissible functions $\phi$; value of $\pi^*$ as defined in (43); expression of update $\delta_j$ in (10) for $\varepsilon = 1/2$; expression of the weight update in (11) (See text for details).

anteed convergence rates. This class, which comprises the popular logistic and squared surrogates [3], match the set of even, twice differentiable *proper scoring rules* [12]. This is a proof of generality of our approach as being "proper" is the bare minimum one can request from a score — it roughly states that forecasting the right output yields the optimal score. Our main theoretical result establishes, for any of these surrogates, convergence rates towards global optimum that surprisingly compete with those known for UNN [5] — thus proving that a complex, time consuming leveraging procedure is not necessary for fast convergence towards the optimum. To the best of our knowledge, these are the first convergence rates under the boosting framework for Newton-Raphson approaches to general surrogate risk minimization, a set whose most prominent member is Gentle Adaboost [3]. The link with balanced convex surrogates optimization allows to show that GNNB equivalently fits class posteriors, and complies with weak universal consistency requirements. Experiments are provided on a dozen domains, including small domains from the UCI repository of machine learning database [13] and large computer vision domains: the Caltech [14] and SUN domains [15]. They display that GNNB outperforms UNN, both in terms of convergence rate and quality of the solutions obtained. They also display that, on large domains for which complex learning approaches like non-linear support vector machines or boosting with deep trees are ruled out for computational considerations, GNNB offers a simple, lightweight and competing alternative to heuristic methods like stochastic gradient descent. Our experiments come with an improvement of GNNB aimed at reducing the weak point represented by the curse of dimensionality for nearest neighbor algorithms on large domains. We provide a low-cost divide-and-conquer scheme which makes a partition of the description variables before running GNNB, and exploits links with density estimation in proper scoring rules to craft, out of all predictions, an aggregated score which is shown experimentally to outperform very significantly the vanilla approach without splitting.

The remaining of the paper is organized as follows: Section 2 provides definitions. Section 3 presents GNNB. Section 4 and Section 5 respectively state and discuss its theoretical properties. Section 6 presents experiments, and Section 7 concludes the paper.

## 2 DEFINITIONS

### 2.1 General setting

Our setting is multiclass, multilabel classification [7]. We have access to an input set of $m$ examples (or prototypes), $\mathcal{S} \doteq \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m\}$. Vector $\boldsymbol{y}_i \in \{-1, 1\}^C$ encodes class memberships, assuming $y_{ic} = 1$ means that observation $\boldsymbol{x}_i$ belongs to class $c$. We let $\mathcal{H} : \mathcal{O} \to \mathbb{R}^C$ denote a classifier, $\mathcal{O}$ being the observations domain to which all $\boldsymbol{x}_i$ belong. The $c^{th}$ coordinate of the output of $\mathcal{H}$, $h_c \doteq \mathcal{H}_c$, is a classifier which segregates observations according to their membership to class $c$. The boosting framework originating in the seminal works of Valiant, has contributed to bring to the fore the interests in learning $\mathcal{H}$ by the minimization of a *total surrogate risk*:

$$\varepsilon_{\mathcal{S}}^{\psi}(\mathcal{H}) \doteq \frac{1}{C}\sum_{c=1}^{C}\varepsilon_{\mathcal{S}}^{\psi}(h_c, c) \ , \quad (1)$$

where

$$\varepsilon_{\mathcal{S}}^{\psi}(h_c, c) \doteq \frac{1}{m}\sum_{i=1}^{m}\psi(y_{ic}h_c(\boldsymbol{x}_i)) \quad (2)$$

is a surrogate risk associated to class $c$, simply named surrogate risk hereafter [3], [11], [16], [7] (and many others). Quantity $y_{ic}h_c(\boldsymbol{x}) \in \mathbb{R}$ is the *edge* of classifier $h$ on example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, for class $c$.

### 2.2 Proper scoring rules and balanced convex surrogates

There exists numerous choices for the (surrogate) *loss* $\psi$. In this subsection, we motivate the analysis of a subset of particular interest, called balanced convex losses [11], [16]. For the sake of clarity, we assume in

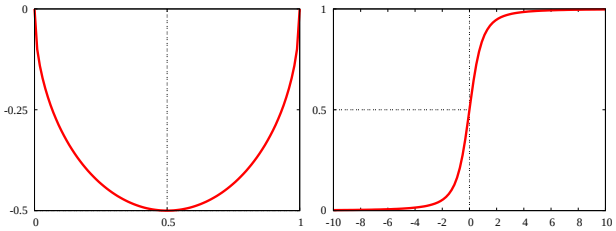


Fig. 1. Plot of permissible $\phi$ of row D in Table 1 (left) and its matching posterior estimate $\hat{p}_{\phi,h}$ as a function of $h \in \mathbb{R}$ (right, see text).

this Subsection that we have two classes ($C = 2$), and reduce the class vector to real $y \in \{-1, 1\}$ encoding membership to a so-called "positive" class ("1"). "$-1$" means observation does not belong to the positive class, or similarly belongs to a "negative" class. In this case, a classifier $h$ outputs a single real value.

More general than the problem of predicting labels is the problem of estimating posteriors [12], [17]: let $p \doteq \hat{p}[y = 1|\boldsymbol{x}]$ define for short the unknown true posterior for observation $\boldsymbol{x}$. The discrepancy between an estimator $\hat{p}$ of $p$ and $p$ is measured by a loss $\ell_{[0,1]}(p\|\hat{p})$. The interval $[0, 1]$ in index recalls that its arguments are probabilities, and "$\|$" means that it is not assumed to be symmetric. There are three requirements one can put on a loss to fit it to statistical requirements of the estimation task while making it suited to convenient algorithmic minimization. The most important one, requirement **R1**, is fundamental in estimation, as it states that $\ell_{[0,1]}$ defines a (strictly) *proper scoring rule*: $0 = \ell_{[0,1]}(p\|p) < \ell_{[0,1]}(p\|q)$, for any $q$ and $p \neq q$ [12], [18], [16], [17]. This requirement is fundamental in that it encourages reliable estimations. Second, requirement **R2** states that the loss is *even* as $\ell_{[0,1]}(p\|\hat{p}) = \ell_{[0,1]}(1 - p\|1 - \hat{p})$, and thus there is no class-dependent mis-estimation cost. For example, predicting $\hat{p} = 1$ (observation always positive) while $p = 0$ (observation always negative) incurs the same loss as predicting $\hat{p} = 0$ while $p = 1$. This is a common assumption in machine learning or classification. Third and last, requirement **R3** states that $\ell_{[0,1]}$ is twice differentiable.

The following Theorem, whose proof can be found in [11], [16], exhibits the true shape of $\ell_{[0,1]}$.

*Theorem 1:* [11], [16] Any loss $\ell_{[0,1]}$ satisfies requirements **R1**–**R3** iff it is a Bregman divergence:

$$\ell_{[0,1]}(p\|q) \quad = \quad D_\phi(p\|q) \ ,$$

for some *permissible* $\phi$.

Theorem 1 makes use of two important definitions: a permissible $\phi$ satisfies: $\phi : [0, 1] \to \mathbb{R}^+$, it is differentiable on $(0, 1)$, strictly convex, twice differentiable on $(0, 1)$ and symmetric around $x = 1/2$. Also, for any strictly convex differentiable $\psi$, the *Bregman divergence* of (strictly convex differentiable) *generator* $\psi$ between $x$ and $x'$ is defined as:

$$D_\psi(x'\|x) \quad \doteq \quad \psi(x') - \psi(x) - (x' - x)\nabla_\psi(x) \ , \quad (3)$$

where "$\nabla$" denotes first order derivative. We are now in a position to define balanced convex losses and relate them to proper scoring rules. Before, let us define the Legendre convex conjugate of any strictly convex differentiable function $\psi$ as $\psi^\star(x) \doteq x\nabla_\psi^{-1}(x) - \psi(\nabla_\psi^{-1}(x))$.

*Definition 1:* [16] Given some permissible $\phi$, the **balanced convex loss** (BCL) with signature $\phi$, $\psi_\phi$, is:

$$\psi_\phi(x) \quad \doteq \quad \frac{\phi^\star(-x) + \phi(0)}{\phi(0) - \phi(1/2)} \ . \quad (4)$$

We then have the following Theorem.

*Theorem 2:* [11], [16] The following identity holds true for any permissible $\phi$ and any classifier $h$:

$$D_\phi(p(y)\|\hat{p}_{\phi,h}(\boldsymbol{x})) \quad = \quad (\phi(0) - \phi(1/2))\psi_\phi(yh(\boldsymbol{x})) \ ,$$

where

$$\hat{p}_{\phi,h}(\boldsymbol{x}) \quad \doteq \quad \nabla_\phi^{-1}(h(\boldsymbol{x})) \in [0, 1] \ , \quad (5)$$

and

$$p(y) \doteq p[y = 1|\boldsymbol{x}] \quad \doteq \quad \begin{cases} 0 & \text{iff} \quad y = -1 \\ 1 & \text{otherwise} \end{cases} \ . \quad (6)$$

Let us call $\hat{p}_{\phi,h}$ the matching posterior estimate for classifier $h$, as it represents an estimate $\hat{p}_{\phi,h}[y = 1|\boldsymbol{x}]$. Figure 1 plots $\hat{p}_{\phi,h}[y = 1|\boldsymbol{x}]$ for choice D in Table 1. It comes from Theorems 1 and 2 that balanced convex losses (for real valued classification) match a wide set of proper scoring rules (for estimation). Thus, they characterize a very important set of losses and we shall see in the following Section how to achieve the optimum of the score through a gentle optimization procedure with nearest neighbor classifiers, with guaranteed rates of convergence.

Table 1 includes the most popular examples of BCLs: squared loss (row A), (normalized) logistic loss (B), binary logistic loss (C), Matsushita's loss (D). Hinge loss (E) is not a BCL, yet it defines the asymptotes of any BCL [11], and its $\phi$ defines the usual empirical loss [11]. Adaboost's exponential loss is not a BCL [3], [7].

We finish by stating some properties of $\phi$ and $\psi_\phi$. Let us assume that

$$\min_{[0,1]} \mathrm{H}_\phi(x) \quad > \quad 0 \ ; \quad (7)$$

this is the case for all examples in Table 1. Otherwise, we may replace $\phi$ by $\phi + \phi_2$ where $\phi_2$ is permissible and meets assumption (7). Since permissibility is closed by linear combinations, function $\phi + \phi_2$ is also permissible and satisfies (7). Since

$$\mathrm{H}_{\psi_\phi}(x) \quad = \quad \frac{1}{(\phi(0) - \phi(1/2)) \times \mathrm{H}_\phi(\nabla_\phi^{-1}(x))} \ ,$$

**Algorithm 1:** Algorithm GENTLE NN BOOSTING, GNNB($\mathcal{S}, \phi, \varepsilon, k$)

**Input**: $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, ..., m, \ \boldsymbol{x_i} \in \mathcal{O}, \ \boldsymbol{y_i} \in \{-1, 1\}^C\}$, permissible $\phi$, $\varepsilon \in (0, 1)$, $k \in \mathbb{N}_*$;

Let $\boldsymbol{\alpha}_j \leftarrow \mathbf{0}, \forall j = 1, 2, ..., m$;
**for** $c = 1, 2, ..., C$ **do**
  Let $\boldsymbol{w} \leftarrow \frac{1}{2(\phi(0) - \phi(1/2))}\mathbf{1}$;
  **for** $t = 1, 2, ..., T$ **do**
    **[I.0]**//Choice of the example to leverage
    Let $j \leftarrow \text{WIC}(\mathcal{S}, \boldsymbol{w})$;
    **[I.1]**//Computation of the gentle leveraging coefficient update, $\delta_j$
    Let

$$\eta(c, j) \leftarrow \sum_{i : j \sim_{\mathcal{S}, k} i} w_{ti} y_{ic} y_{jc} \ ; \quad (9)$$

$$\delta_j \leftarrow \frac{2(1 - \varepsilon)\eta(c, j)}{\mathrm{H}^*_{\psi_\phi} n_j} \ , \text{ with } n_j \doteq |\{i : j \sim_k i\}| \ ; \quad (10)$$

    **[I.2]**//Weights update
    $\forall i : j \sim_k i$, let

$$w_i \leftarrow \frac{\nabla_\phi^{-1}\left(-\delta_j y_{ic} y_{jc} + \nabla_\phi((\phi(0) - \phi(1/2))w_i)\right)}{\phi(0) - \phi(1/2)} \ ; \quad (11)$$

    // we have $w_i \in \left[0, (\phi(0) - \phi(1/2))^{-1}\right]$
    **[I.3]**//Leveraging coefficient update
    Let $\alpha_{jc} \leftarrow \alpha_{jc} + \delta_j$;

**Output**: $\mathcal{H}(\boldsymbol{x}) \doteq \sum_{j \sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j$

assumption (7) implies:

$$\mathrm{H}^*_{\psi_\phi} \doteq \sup_{\mathbb{R}} \mathrm{H}_{\psi_\phi}(x) \quad \ll \quad \infty \ , \quad (8)$$

and in fact $\mathrm{H}^*_{\psi_\phi} = \mathrm{H}_{\psi_\phi}(0)$ for all examples in Table 1, and is very small (Cf column $\pi^*$, (43) and Section 4). The following Lemma states properties shown in [16].

*Lemma 1:* [16] For any permissible $\phi$, the following properties hold true: $\phi^\star(x) = \phi^\star(-x) + x, \forall x$; $\nabla_{\psi_\phi}(0) < 0$, $\psi_\phi(0) = 1$, $\mathrm{im}(\psi_\phi) \subseteq \mathbb{R}^+$.

### 2.3 Empirical risk and its minimization

Lemma 1 implies that surrogate risk minimization may be used as an approximate primer to the minimization of the empirical risk, as indeed the total surrogate risk (1) is an upperbound of the empirical (Hamming) risk [7]:

$$\varepsilon^{\mathrm{H}}_{\mathcal{S}}(\mathcal{H}) \doteq \frac{1}{C}\sum_{c=1}^{C} \varepsilon^{0/1}_{\mathcal{S}}(h_c, c) \quad \leq \quad \frac{1}{\psi(0)}\varepsilon^{\psi}_{\mathcal{S}}(\mathcal{H}) \ , \quad (12)$$

where

$$\varepsilon^{0/1}_{\mathcal{S}}(h_c, c) \quad \doteq \quad \frac{1}{m}\sum_{i=1}^{m} \mathrm{I}[y_{ic} h_c(\boldsymbol{x}_i) < 0] \quad (13)$$

is the usual empirical risk associated to class $c$. To quantify the performance of the best possible classifier, we respectively define:

$$(\varepsilon^{\psi_\phi}_{\mathcal{S}})^*_c \quad \doteq \quad \inf_h \varepsilon^{\psi_\phi}_{\mathcal{S}}(h, c) \ , \quad (14)$$

$$(\varepsilon^{0/1}_{\mathcal{S}})^*_c \quad \doteq \quad \inf_h \varepsilon^{0/1}_{\mathcal{S}}(h, c) \ , \quad (15)$$

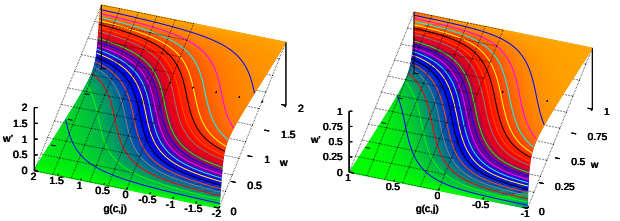


Fig. 2. Weight update $w'$ computed as a function of $w$ and $g(c, j) \doteq \eta(c, j)/n_j$, when $y_{ic} y_{jc} = 1$ and $\varepsilon = 1/2$ (see Table 1). The corresponding BCLS are the binary logistic loss (left) and Matsushita's loss (right). The black grid depicts the plane of equation $w' = w$.

as the respective Bayes surrogate risks and Bayes empirical risks for class $c$. Averaging these expressions following (1) and (12), we respectively define $(\varepsilon^{\psi_\phi}_{\mathcal{S}})^*$ and $(\varepsilon^{0/1}_{\mathcal{S}})^*$ as the optimal total surrogate risk and empirical (Hamming) risk on $\mathcal{S}$. As a last remark, our minimization problems on the learning sample may be useful as well to minimize the *true (surrogate) risks*, that is, expectations of (1, 12) in generalization, according to some unknown distribution from which $\mathcal{S}$ is supposed i.i.d. sampled. We refer to [9], [19] and the references therein for details, not needed here.

## 3 GENTLE BOOSTING FOR NN RULES

The nearest neighbors (NNs) rule belongs to the oldest, simplest and still most widely studied classification algorithms [20]. It relies on a non-negative real-valued "distance" function. This function is defined on domain $\mathcal{O}$ and measures how much two observations differ from each other. This dissimilarity function thus may not necessarily satisfy the triangle inequality of metrics. For the sake of readability, we let $j \sim_k \boldsymbol{x}$ denote the assertion that example $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, or simply example $j$, belongs to the $k$ NNs of observation $\boldsymbol{x}$. We shall abbreviate $j \sim_k \boldsymbol{x}_i$ by $j \sim_k i$ — in this case, we say that example $i$ belongs to the *inverse neighborhood* of example $j$. To classify an observation $\boldsymbol{x} \in \mathcal{O}$, the $k$-NN rule $\mathcal{H}$ over $\mathcal{S}$ computes the sum of class vectors of its nearest neighbors, that is: $\mathcal{H}(\boldsymbol{x}) = \sum_{j \sim_k \boldsymbol{x}} \mathbf{1} \circ \boldsymbol{y}_j$, where $\circ$ is the Hadamard product[1]. $\mathcal{H}$ predicts that $\boldsymbol{x}$ belongs to each class whose corresponding coordinate in the final vector is positive. A *leveraged* $k$-NN rule generalizes this to:

$$\mathcal{H}(\boldsymbol{x}) \quad = \quad \sum_{j \sim_k \boldsymbol{x}} \boldsymbol{\alpha}_j \circ \boldsymbol{y}_j \ , \quad (16)$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^C$ is a leveraging vector for the classes in $\boldsymbol{y}_j$. Leveraging approaches to nearest neighbors are not new [21], [22], yet to the best of our knowledge no convergence rates were known, at least until the algorithm UNN [5], [6]. Algorithm 1 presents our gentle boosting algorithm for the nearest neighbor

1. Coordinate-wise.

rules, GNNB. It differs with UNN on the key part of (16): the computation and update of the leveraging vectors. Instead of the repetitive solving of nonlinear equations — time consuming and with the risk, for approximations, of lying outside the boosting regime —, we prefer a simple scheme linear on the *weighted edge* $\eta(c,j)$ (see Algorithm 1). The scheme of UNN [5], [6] is nonlinear in this parameter. Our updates also depend on integer $n_j$, the cardinality of the inverse neighborhood of example $j$, where $|.|$ denotes the cardinality (see Algorithm 1). Table 1 gives the expressions of the weight update (11) for various choices of permissible functions $\phi$, and the expression of $\delta_j$ for the particular choice $\varepsilon = 1/2$. Figure 2 plots examples of the weight update (11). The ranges of values, used in Figure 2, are respectively

$$\left[-\frac{1}{\phi(0)-\phi(1/2)}, \frac{1}{\phi(0)-\phi(1/2)}\right]$$

for $g(c,j)$, and

$$\left[0, \frac{1}{\phi(0)-\phi(1/2)}\right]$$

for $w$ and $w'$. The two plots, similar, exemplify two important remarks valid for any BCL. First, when classes match for example $i$ and $j$, the weight of example $i$ decreases iff $\delta_j > 0$. This is a common behavior for boosting algorithms. Second, the regime of weight variations for extreme values of $g(c,j)$ appear to be very important, despite the fact that leveraging update $\delta_j$ is linear in the weighted edge. Thus, "gentle" updates do not prevent significant variations in weights.

## 4 Properties of GNNB

### 4.1 GNNB is Newton-Raphson

Our first result establishes that GNNB performs Newton-Raphson updates to optimize its surrogate risk, like Gentle Adaboost [3]. If we pick example $i$ in the inverse neighborhood of example $j$ to be updated for class $c$, we have $\partial\psi_\phi(y_{ic}h_c(\boldsymbol{x}_i))/\partial\delta_j = -w_i y_{ic} y_{jc}$, and $\partial^2\psi_\phi(y_{ic}h_c(\boldsymbol{x}_i))/\partial\delta_j^2 = \mathrm{H}_{\psi_\phi}(y_{ic}h_c(\boldsymbol{x}_i))$, so that the Newton-Raphson update for $\delta_j$ reads:

$$\delta_j \quad \leftarrow \quad \rho \times \frac{\eta(c,j)}{\sum_{i:j\sim_{\mathcal{S},k} i} \mathrm{H}_{\psi_\phi}(y_{ic}h_c(\boldsymbol{x}_i))} \ , \qquad (17)$$

for some small learning rate $\rho$, typically with $0 < \rho \leq 1$. Comparing with (10), we get the following result.

*Theorem 3:* GNNB uses adaptive Newton-Raphson steps to minimize the surrogate risk at hand, $\varepsilon_{\mathcal{S}}^{\psi_\phi}$, with adaptive learning rate $\rho \doteq \rho(c,j,\varepsilon)$:

$$\rho(c,j,\varepsilon) \quad = \quad \frac{2(1-\varepsilon)\sum_{i:j\sim_{\mathcal{S},k} i} \mathrm{H}_{\psi_\phi}(y_{ic}h_c(\boldsymbol{x}_i))}{\mathrm{H}^*_{\psi_\phi} n_j} \qquad (18)$$

Furthermore,

$$0 < \quad \rho(c,j,\varepsilon) \quad < 2(1-\varepsilon) \ .$$

The Newton-Raphson flavor of GNNB might be useful to prove its convergence to the optimum of the surrogate risk at hand ($\varepsilon_{\mathcal{S}}^{\psi_\phi}$), yet the original boosting theory is more demanding than "mere" convergence to global optimum: it requires guaranteed convergence rates under weak assumptions about each iteration.

### 4.2 GNNB boosts the surrogate risks

We consider the following *Weak Learning Assumption* about GNNB:

(**WLA**) There exist constants $\varrho > 0, \vartheta > 0$ such that at any iterations $c, t$ of GNNB, index $j$ returned by WIC is such that the following holds:

$$\frac{\sum_{i:j\sim_{\mathcal{S},k} i} w_i}{n_j} \geq \frac{\varrho}{\phi(0)-\phi(1/2)} \ ; \qquad \textbf{(i)}$$

$$|\hat{p}_{\boldsymbol{w}}[y_{jc} \neq y_{ic} | j \sim_{\mathcal{S},k} i] - 1/2| \geq \vartheta \ . \qquad \textbf{(ii)}$$

Requirement (ii) corresponds to the usual weak learning assumption of boosting [11], [16], [7]: it postulates that the current *normalized* weights in the inverse neighborhood of example $j$ authorize a classification different from random by at least $\vartheta$. GNNB uses unnormalized weights that satisfy $(1/n_j)\sum_{i:j\sim_{\mathcal{S},k} i} w_i \in [0, 1/(\phi(0)-\phi(1/2))]$: requirement (i) thus implies that the unnormalized weights in the inverse neighborhood must not be too small. Intuitively, such a condition is necessary as unnormalized weights of minute order would not necessarily prevent (ii) to be met, but would impair the convergence of GNNB given the linear dependence of $\delta_j$ in the unnormalized weights. Notice also that unnormalized weights are all the smaller as examples receive the right labels: the fact that requirement (i) becomes harder to be met simply means that GNNB approaches the optimum sought. At the beginning of GNNB, the initialization with the null leveraging vectors ($\boldsymbol{\alpha}_j = \mathbf{0}, \forall j$) guarantees that we can pick in (i) $\varrho = 1/2$ everywhere.

The analysis we carry out is a bit more precise than usual boosting results: instead of giving, under the **WLA**, a lowerbound on the number of iterations needed to drive down the surrogate or empirical risks down some user-fixed threshold $\tau$, we rather provide a lowerbound on the total number of *weight updates*, for each class $c$. This number, $\ell(T,c)$, integrates the total number of boosting iterations and the size of inverse neighborhoods used. It is important to integrate these sizes since there is obviously a big difference for convergence between leveraging an example which votes for many others in "dense" parts of the data, and leveraging one which votes for none. Our main result is split in two. The first focuses on the surrogate risk, the second on the empirical risk. Let us first

define:

$$\phi_c(\mathcal{S}) \doteq \sum_{\boldsymbol{x}} \hat{p}_{\mathcal{S}}[\boldsymbol{x}]\phi(\hat{p}_{\mathcal{S}}[y_c = 1|\boldsymbol{x}]) \ , \quad (19)$$

$$\Delta_\phi(\mathcal{S}, \tau, c) \doteq \phi_c(\mathcal{S}) - ((1-\tau)\phi(1/2) + \tau\phi(0)) \quad (20)$$

$$\Delta'_\phi(\mathcal{S}, \tau, c) \doteq \phi_c(\mathcal{S}) - \phi\left(\frac{1-\tau}{2}\right) \ . \quad (21)$$

$\Delta_\phi(\mathcal{S}, \tau, c)$ and $\Delta'_\phi(\mathcal{S}, \tau, c)$ are differences between average values of $\phi$ taking values within $\pm(\phi(0) - \phi(1/2))$. We are now ready to state our main result on GNNB.

*Theorem 4:* Assume the **WLA** holds, and let $\tau \in [0, 1]$. Suppose we run GNNB so that, $\forall c$, $\ell(T, c)$ meets:

$$\ell(T, c) \geq \frac{\Delta_\phi(\mathcal{S}, \tau, c)(\phi(0) - \phi(1/2))H^*_{\psi_\phi}}{8\varepsilon(1-\varepsilon)\vartheta^2\varrho^2} \times m \quad (22)$$

Then the leveraged $k$-NN $\mathcal{H}$ learned by GNNB satisfies:

$$\varepsilon^{\psi_\phi}_{\mathcal{S}}(\mathcal{H}) \leq (\varepsilon^{\psi_\phi}_{\mathcal{S}})^* + \tau \ . \quad (23)$$

*Proof:* Our main objective is to craft a negative upperbound for the variation of the surrogate risk at hand (2) between two successive iterations, say $t$ and $t+1$. To keep references clear, we replace the index $j$ of the example returned by WIC by $e(t)$. We have:

$$\begin{aligned}
&\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc}, c) \\
&= \frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i} \psi_\phi(y_{ic}h_{(t+1)c}(\boldsymbol{x})) \\
&\quad -\frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i} \psi_\phi(y_{ic}h_{tc}(\boldsymbol{x})) \\
&= \frac{1}{m}\left\{\sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}(0\|\nabla^{-1}_{\tilde{\psi}_\phi}(-y_{ic}h_{(t+1)c}(\boldsymbol{x}))) \right. \\
&\quad \left. - \sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}(0\|\nabla^{-1}_{\tilde{\psi}_\phi}(-y_{ic}h_{tc}(\boldsymbol{x})))\right\} \ , \quad (24)
\end{aligned}$$

where $\tilde{\psi}_\phi(x) \doteq (\psi_\phi)^\star(-x)$ and (24) comes from Lemma 1 in [16]. Now, we obtain $\nabla_{\psi_\phi}(x) = -\nabla^{-1}_\phi(-x)/(\phi(0) - \phi(1/2))$, and so

$$\begin{aligned}
\nabla^{-1}_{\psi_\phi}(x) &= \nabla_{(\psi_\phi)^\star}(x) \\
&= -\nabla_\phi((\phi(1/2) - \phi(0))x) \ ,
\end{aligned}$$

which yields

$$\tilde{\psi}_\phi(x) = -\frac{\phi\left((\phi(0) - \phi(1/2))x\right)}{\phi(0) - \phi(1/2)} \ . \quad (25)$$

Furthermore,

$$\begin{aligned}
&\nabla^{-1}_{\tilde{\psi}_\phi}(-y_{ic}h_{(t+1)c}(\boldsymbol{x}_i)) \\
&= \frac{1}{\phi(0) - \phi(1/2)}\nabla^{-1}_\phi(-\delta_{e(t)}y_{ic}y_{e(t)c} - y_{ic}h_{tc}(\boldsymbol{x}_i)) \\
&= w_{(t+1)i} \ , \quad (26)
\end{aligned}$$

and $\nabla^{-1}_{\tilde{\psi}_\phi}(-y_{ic}h_{tc}(\boldsymbol{x})) = w_{ti}$ as well, so that, taking into account (25) and (26), we obtain that we can simplify (24) as follows:

$$\begin{aligned}
&\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc}, c) \\
&= \frac{1}{m}\left\{\sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}(0\|w_{(t+1)i}) \right. \\
&\quad \left. - \sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}(0\|w_{ti})\right\} \\
&= -\frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i} \left\{\tilde{\psi}_\phi(w_{(t+1)i}) - \tilde{\psi}_\phi(w_{ti}) \right. \\
&\quad \left. -w_{(t+1)i}\nabla_{\tilde{\psi}_\phi}(w_{(t+1)i}) + w_{ti}\nabla_{\tilde{\psi}_\phi}(w_{ti})\right\} \\
&= -\frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i} \left\{\tilde{\psi}_\phi(w_{(t+1)i}) - \tilde{\psi}_\phi(w_{ti}) \right. \\
&\quad +w_{(t+1)i}\left[\delta_{e(t)}y_{ic}y_{e(t)c} - \nabla_{\tilde{\psi}_\phi}(w_{ti})\right] \\
&\quad \left. +w_{ti}\nabla_{\tilde{\psi}_\phi}(w_{ti})\right\} \\
&= -\frac{1}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i} D_{\tilde{\psi}_\phi}(w_{(t+1)i}\|w_{ti}) \\
&\quad -\frac{\delta_{e(t)}}{m}\sum_{i:e(t)\sim_{\mathcal{S},k}i} w_{(t+1)i}y_{ic}y_{e(t)c} \ . \quad (27)
\end{aligned}$$

We now work on lowerbounding the divergence term above. For this objective, we prove a simple but crucial property for $\psi_\phi$, of independent interest. Following [23], we say that a differentiable function $\psi$ is $\omega$ *strongly smooth* iff there exists some $\omega > 0$ such that,

$$D_\psi(x'\|x) \leq \frac{\omega}{2}(x' - x)^2 \ , \forall x, x' \ .$$

*Lemma 2:* For any permissible $\phi$, $\psi_\phi$ is $\mathrm{H}^*_{\psi_\phi}$ strongly smooth, where $\mathrm{H}^*_{\psi_\phi}$ is defined in (8).

*Proof:* Taylor-Lagrange remainder brings that there exists some $x'' \in (x, x')$ such that

$$\begin{aligned}
D_{\psi_\phi}(x'\|x) &= \frac{1}{2} \times (x - x')^2\mathrm{H}_{\psi_\phi}(x'') \\
&\leq \frac{1}{2} \times (x - x')^2\mathrm{H}^*_{\psi_\phi} \ ,
\end{aligned}$$

the inequality coming from (8). This ends the proof of the Lemma. □

It comes from [23] that $(\psi_\phi)^\star$ is $(\mathrm{H}^*_{\psi_\phi})^{-1}$ strongly convex, that is, $(\psi_\phi)^\star(w) - (2\mathrm{H}^*_{\psi_\phi})^{-1}w^2$ is convex. Equivalently,

$$\tilde{\psi}_\phi(w) - \frac{1}{2\mathrm{H}^*_{\psi_\phi}}w^2 \text{ is convex.} \quad (28)$$

Any convex function $\varphi$ satisfies $\varphi(w') \geq \varphi(w) + \nabla_\varphi(w)(w' - w), \forall w, w'$. We apply this inequality taking as $\varphi$ the function in (28), $w = w_{ti}$ and $w' = w_{(t+1)i}$. After simplification, we sum for each $i$ such that

$e(t) \sim_{\mathcal{S},k} i$, and obtain:

$$\sum_{i:e(t)\sim_{\mathcal{S},k} i} D_{\tilde{\psi}_\phi}\left(w_{(t+1)i}||w_{ti}\right) \geq \frac{1}{2\mathrm{H}^*_{\psi_\phi}} \sum_{i:e(t)\sim_{\mathcal{S},k} i} \left(w_{(t+1)i} - w_{ti}\right)^2 . \quad (29)$$

Finally, Cauchy-Schwartz inequality yields:

$$\sum_{i:e(t)\sim_{\mathcal{S},k} i} \left(y_{ic}y_{e(t)c}\right)^2 \sum_{i:e(t)\sim_{\mathcal{S},k} i} \left(w_{(t+1)i} - w_{ti}\right)^2 \geq \left(\sum_{i:e(t)\sim_{\mathcal{S},k} i} y_{ic}y_{e(t)c}(w_{(t+1)i} - w_{ti})\right)^2 . \quad (30)$$

Fix for short $u \doteq \sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{(t+1)i}y_{ic}y_{e(t)c}$. Plugging altogether (27), (29) and (30), we obtain the following upperbound for $\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc}, c)$:

$$\begin{aligned}
&\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc}, c) \\
&\leq -\frac{\left(u - \sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{ti}y_{ic}y_{e(t)c}\right)^2}{2H^*_{\psi_\phi} m \sum_{i:e(t)\sim_{\mathcal{S},k} i} \left(y_{ic}y_{e(t)c}\right)^2} - \frac{\delta_{e(t)}}{m}u \\
&= \frac{\Delta_t(u)}{m} . \quad (31)
\end{aligned}$$

$\Delta_t(u)$ takes its maximum value for $u = u^* \doteq \eta(c, e(t)) - H^*_{\psi_\phi} n_{e(t)}\delta_{e(t)}$, for which we have:

$$\Delta_t(u^*) = \frac{H^*_{\psi_\phi} n_{e(t)}\delta_{e(t)}}{2}\left(\delta_{e(t)} - \frac{2\eta(c, e(t))}{H^*_{\psi_\phi} n_{e(t)}}\right) .$$

Suppose we pick $\delta_{e(t)}$ as in (10), that is, $\delta_{e(t)} = 2(1-\varepsilon)\eta(c, e(t))(H^*_{\psi_\phi} n_{e(t)})^{-1}$, for $\varepsilon \in (0, 1)$. This choice yields:

$$\begin{aligned}
\Delta_t(u) &\leq \Delta_t(u^*) \\
&= -2\varepsilon(1-\varepsilon)\frac{\eta^2(c, e(t))}{H^*_{\psi_\phi} n_{e(t)}} . \quad (32)
\end{aligned}$$

We now show that the **WLA** implies a strictly positive lowerbound on the absolute value of the edge $\eta(c, e(t))$. Defining $\mathrm{I}[.]$ as the indicator function, We have:

$$\begin{aligned}
&\hat{p}_{\boldsymbol{w}_t}[y_{e(t)c} \neq y_{ic} | e(t) \sim_{\mathcal{S},k} i] \\
&= \frac{\sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{ti}\mathrm{I}\left[y_{ic}y_{e(t)c} = -1\right]}{\sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{ti}} \\
&= \frac{\sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{ti}\left(1 - y_{ic}y_{e(t)c}\right)}{2\sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{ti}} \\
&= \frac{1}{2} - \frac{\eta(c, e(t))}{2\sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{ti}} .
\end{aligned}$$

Using statement (ii) in the **WLA** with this last equality brings

$$|\eta(c, e(t))| \geq 2\vartheta \sum_{i:e(t)\sim_{\mathcal{S},k} i} w_{ti} .$$

Using statement (i) in the **WLA**, we finally obtain:

$$|\eta(c, e(t))| \geq 2\frac{\vartheta\varrho n_{e(j)}}{\phi(0) - \phi(1/2)} . \quad (33)$$

Plugging (33) into (32), and the resulting inequality into (31), we obtain:

$$\begin{aligned}
&\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{(t+1)c}, c) - \varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{tc}, c) \\
&\leq -8\varepsilon(1-\varepsilon)\frac{n_{e(t)}\vartheta^2\varrho^2}{mH^*_{\psi_\phi}(\phi(0) - \phi(1/2))^2} . \quad (34)
\end{aligned}$$

At the initialization, all leveraging coefficients $\boldsymbol{\alpha}_j$ equal the null vector, and so the corresponding surrogate risk equals $\psi_\phi(0)$. To guarantee that $\varepsilon^{\psi_\phi}_{\mathcal{S}}(h_{Tc}, c) \leq (\varepsilon^{\psi_\phi}_{\mathcal{S}})^*_c + \tau$ under the **WLA**, for some $\tau \in [0, \psi_\phi(0)]$, it is thus sufficient to have:

$$\begin{aligned}
&\sum_{t=1}^{T} n_{e(t)} \\
&\geq \frac{(\psi_\phi(0) - (\varepsilon^{\psi_\phi}_{\mathcal{S}})^*_c - \tau)H^*_{\psi_\phi}(\phi(0) - \phi(1/2))^2}{8\varepsilon(1-\varepsilon)\vartheta^2\varrho^2} \times m .
\end{aligned}$$

This inequality leads to the statement of the Theorem, provided we remark the three following facts. The first one is proven in the following Lemma, of independent interest.

*Lemma 3:* We have

$$(\varepsilon^{\psi_\phi}_{\mathcal{S}})^*_c = \frac{\phi(0) - \phi_c(\mathcal{S})}{\phi(0) - \phi(1/2)} ,$$

where $\phi_c(\mathcal{S})$ is defined in (19).

*Proof:* From Lemma 1, we have the property $\nabla^{-1}_\phi(x) = 1 - \nabla^{-1}_\phi(-x)$, with which we obtain after few derivations that:

$$\arg\min_h \varepsilon^{\psi_\phi}_{\mathcal{S}_{\boldsymbol{x}}}(h, c) = \nabla_\phi(\hat{p}_{\mathcal{S}}[y_c = 1|\boldsymbol{x}]) , \quad (35)$$

where $\mathcal{S}_{\boldsymbol{x}}$ is the subset of $\mathcal{S}$ whose observations match $\boldsymbol{x}$. Then, we compute $\varepsilon^{\psi_\phi}_{\mathcal{S}}(h, c)$ with this value for $h$, which, after simplification using Legendre conjugates, brings

$$\mathbf{E}_{\mathcal{S}_{\boldsymbol{x}}}[\psi_\phi(y_{ic}h(\boldsymbol{x}))] = \frac{\phi(0) - \phi(\hat{p}_{\mathcal{S}}[y_c = 1|\boldsymbol{x}])}{\phi(0) - \phi(1/2)} .$$

Finally, we average this over all distinct observations in $\mathcal{S}$ to obtain Lemma 3. □

The last two facts that lead to the statement of the Theorem are simpler: we indeed have $\sum_{t=1}^{T} n_{e(t)} = \ell(T, c)$, and $\psi_\phi(0) = (\phi(0) - \phi(\nabla^{-1}_\phi(0)))/(\phi(0) - \phi(1/2)) = 1$. This concludes the proof of Theorem 4. □

### 4.3 GNNB boosts the empirical risk

The bound of Theorem 4 translates to a similar bound for the empirical risk.

*Corollary 1:* Assume the **WLA** holds, and let $\tau \in [0, 1]$. Suppose we run GNNB so that, $\forall c$, $\ell(T, c)$ meets:

$$\ell(T, c) \geq \frac{\Delta'_\phi(\mathcal{S}, \tau, c)(\phi(0) - \phi(1/2))H^*_{\psi_\phi}}{8\varepsilon(1-\varepsilon)\vartheta^2\varrho^2} \times m \quad (36)$$

Then the leveraged $k$-NN $\mathcal{H}$ learned by GNNB satisfies:

$$\varepsilon_{\mathcal{S}}^{\mathrm{H}}(\mathcal{H}) \quad \leq \quad (\varepsilon_{\mathcal{S}}^{\mathrm{H}})^* + \tau \ . \tag{37}$$

*Proof:* Following [9], let us define

$$\begin{aligned} H(\epsilon) &\doteq \inf_{\delta\in\mathbb{R}} \{\epsilon\psi_\phi(\delta) + (1-\epsilon)\psi_\phi(-\delta)\} \ , \\ \psi_{\text{BJM}}(\epsilon') &\doteq \psi_\phi(0) - H\left(\frac{1+\epsilon'}{2}\right) \ , \end{aligned}$$

with $\epsilon \in [0,1]$ and $\epsilon' \in [-1,1]$. We have:

$$\begin{aligned} H(\epsilon) &= \inf_{\delta\in\mathbb{R}} \left\{ \frac{\epsilon\phi^\star(-\delta) + (1-\epsilon)\phi^\star(\delta) + \phi(0)}{\phi(0) - \phi(1/2)} \right\} \\ &= \inf_{\delta\in\mathbb{R}} \left\{ \frac{\phi^\star(\delta) - \epsilon\delta + \phi(0)}{\phi(0) - \phi(1/2)} \right\} \qquad (38) \\ &= \frac{\phi^\star(\nabla_\phi(\epsilon)) - \epsilon\nabla_\phi(\epsilon) + \phi(0)}{\phi(0) - \phi(1/2)} \\ &= \frac{-\phi^{\star\star}(\epsilon) + \phi(0)}{\phi(0) - \phi(1/2)} = \frac{\phi(0) - \phi(\epsilon)}{\phi(0) - \phi(1/2)} \ . \qquad (39) \end{aligned}$$

Here, (38) follows from Lemma 1, and (39) follows from the fact that $\phi$ is convex and lower semicontinuous. We thus have:

$$\psi_{\text{BJM}}(\epsilon') \quad = \quad \frac{\phi((1-\epsilon')/2) - \phi(1/2)}{\phi(0) - \phi(1/2)} \ . \tag{40}$$

It is proven in [9], Theorem 1, that:

$$\psi_{\text{BJM}}\left(\varepsilon_{\mathcal{S}}^{0/1}(h_c,c) - (\varepsilon_{\mathcal{S}}^{0/1})_c^*\right) \quad \leq \quad \varepsilon_{\mathcal{S}}^{\psi_\phi}(h_c,c) - (\varepsilon_{\mathcal{S}}^{\psi_\phi})_c^* \ .$$

The argument of $\psi_{\text{BJM}}$ is in $[0,1]$. On this interval, $\psi_{\text{BJM}}$ admits an inverse because $\phi$ admits an inverse on $[0,1/2]$. To ensure

$$\varepsilon_{\mathcal{S}}^{0/1}(h_c,c) \quad \leq \quad (\varepsilon_{\mathcal{S}}^{0/1})_c^* + \tau' \ ,$$

it is thus equivalent to ensure

$$\varepsilon_{\mathcal{S}}^{\psi_\phi}(h_c,c) - (\varepsilon_{\mathcal{S}}^{\psi_\phi})_c^* \quad \leq \quad \psi_{\text{BJM}}(\tau') \ .$$

There remains to combine (40) and (22) to obtain the statement of Corollary 1. □

### 4.4 GNNB is universally consistent

We analyze GNNB in the setting where WIC yields the leveraging of a subset of $m' < m$ examples out of the $m$ available in $\mathcal{S}$. This setting is interesting because it covers the optimization of GNNB in which we repeatedly leverage the most promising example, for example from the standpoint of $|\delta_j|$. We call GNNB$^*$ this variation of GNNB. We assume that $\mathcal{S}$ is sampled i.i.d. according to some fixed density $D$. Let $v$ be the number of Voronoi cells in the $k$-NN rule built from this subsample of $m'$ examples. The Voronoi cells, say $\mathcal{V}_l$ for $l = 1,2,...,v$, partition $\mathcal{S}_m$ into subsets on which the output $h$ of GNNB is the same, say $h_l$. We drop the $c$ index as we make the analysis for a single class and its associated empirical risk $\varepsilon_{\mathcal{S}}^{0/1}(h) \doteq \varepsilon_{\mathcal{S}}^{0/1}(h_c,c)$, after which the extension to the Hamming risk poses no problem as it is the average of the per-class empirical risks. The empirical minimization of surrogate BCL $\psi_\phi$ in an NN approach amounts to a maximum likelihood fitting of class posteriors. Indeed, the empirical risk $\varepsilon_{\mathcal{S}}^{0/1}(h)$ in (13) can be expressed as, letting $\xi \doteq (\phi(0) - \phi(1/2))^{-1}$:

$$\begin{aligned} \varepsilon_{\mathcal{S}}^{0/1}(h) &\doteq \sum_{\mathcal{S}_m} \hat{p}[(\boldsymbol{x},y_c)]\psi_\phi(y_c h(\boldsymbol{x})) \\ &= \sum_{l=1}^{v} \sum_{\mathcal{S}_m\cap\mathcal{V}_l} \hat{p}[(\boldsymbol{x},y_c)]\psi_\phi(y_c h(\boldsymbol{x})) \\ &= \sum_{l=1}^{v} \hat{p}[\mathcal{V}_l] \sum_{\mathcal{S}_m\cap\mathcal{V}_l} \frac{\hat{p}[(\boldsymbol{x},y_c)]}{\hat{p}[\mathcal{V}_l]}\psi_\phi(y_c h(\boldsymbol{x})) \\ &= \xi \sum_{l=1}^{v} \hat{p}[\mathcal{V}_l] \sum_{\mathcal{S}_m\cap\mathcal{V}_l} \frac{\hat{p}[(\boldsymbol{x},y_c)]}{\hat{p}[\mathcal{V}_l]} D_\phi(p(y_c)\|\hat{p}_{\phi,h}(\boldsymbol{x})) \\ &= \xi \sum_{l=1}^{v} \hat{p}[\mathcal{V}_l] \underbrace{\sum_{\mathcal{S}_m\cap\mathcal{V}_l} \frac{\hat{p}[(\boldsymbol{x},y_c)]}{\hat{p}[\mathcal{V}_l]} D_\phi(p(y_c)\|\hat{p}_l)}_{\varepsilon_l} \ . \qquad (41) \end{aligned}$$

In these identities, all $\hat{p}$ are estimates built from $\mathcal{S}_m$ and $\hat{p}_{\phi,h}(\boldsymbol{x})$ is defined in (5). We have used notation $\hat{p}[\mathcal{V}_l] \doteq \sum_{\mathcal{S}_m\cap\mathcal{V}_l} \hat{p}[(\boldsymbol{x},y_c)]$ The penultimate identity follows from Theorem 2 (and notations used therein). The last identity exploits the fact that, in NN rules, the estimate $\hat{p}_{\phi,h}(\boldsymbol{x})$ is constant over each voronoi cell, so we write it $\hat{p}_l$ for cell $\mathcal{V}_l$. The leveraging in GNNB minimizes (41) from Theorem 3 and the fact that $h_l$ satisfies $\hat{p}_{\phi,h}(\boldsymbol{x}) = \nabla_\phi^{-1}(h_l)$ from (5). Finding the population minimizers $\hat{p}_l$ in $\varepsilon_l$ of (41) is simple as the right-population minimizer of any Bregman divergence is always the arithmetic average [24] (Proposition 1); that is, at the minimum of each $\varepsilon_l$ in (41), $\hat{p}_l = \hat{p}_l^*$ with:

$$\begin{aligned} \hat{p}_l^* &\doteq \sum_{\mathcal{S}_m\cap\mathcal{V}_l} \frac{\hat{p}[(\boldsymbol{x},y_c)]}{\hat{p}[\mathcal{V}_l]} p(y_c) \qquad (42) \\ &= \hat{p}[y_c = +1|\mathcal{V}_l] \ . \end{aligned}$$

Since each example in $m'$ may contribute to more than one Voronoi cell, the achievable minimum of (41) may not exactly satisfy $\hat{p}_l = \hat{p}_l^*$ for each $l$ if $m' \ll \infty$. However, as $m' \to \infty$ and provided $k/m' \to 0$, the distance between each point and all its $k$-NN vanishes [20] (Lemma 5.1), which means, provided the class posteriors are uniformly continuous, that each leveraged example will contribute to Voronoi cells in which posteriors asymptotically converge to the same value, ensuring this time $\hat{p}_l \to \hat{p}_l^*$ as $m'/m \to 0$. Corollary 6.2 in [20] makes that a sufficient condition for the (weak) universal consistency of GNNB with respect to class $c$, and by extension to all classes through the Hamming risk, as stated in the following Lemma. We also note that the Lemma applies without change to UNN [5].

*Lemma 4:* Provided $T, k \to \infty$, $k/m' \to 0$, $m'/m \to 0$, GNNB$^*$ is universally consistent: its expected Hamming risk over i.i.d. size-$m$ samples converges to the Hamming risk of Bayes rule.

## 5 Discussion

The fact that we do not normalize permissible functions, *i.e.* typically ensuring $\phi(1/2) = 1$ and $\phi(0) = 0$, is due to the fact that normalization would reduce the number of BCL that can be generated. For example, out of the two in rows B and C in Table 1, the classical form of the logistic loss in B would disappear. The bounds in (22) and (36) advocate for a very intuitive and computationally affordable implementation of WIC: since the number of examples *leveraged* equals, on average, $\ell(T,c)/k$, we should put emphasis on leveraging examples with large inverse neighborhoods.

Our results call for several technical comparisons between GNNB, UNN and mathematical greedy algorithms [2]. Let us define:

$$\pi^* \doteq \frac{(\phi(0) - \phi(1/2))^2 H^*_{\psi_\phi}}{2} , \quad (43)$$

and let us respectively define $\pi(\varepsilon)$ and $\pi'(\varepsilon)$ the terms factoring $m(\vartheta^2\varrho^2)^{-1}$ in (22) and (36). Because

$$\Delta_\phi(\mathcal{S},\tau,c) \leq \Delta'_\phi(\mathcal{S},\tau,c) \leq \phi(0) - \phi(1/2) ,$$

it comes $\pi(1/2) \leq \pi'(1/2) \leq \pi^*$. Table 1 provides examples of values for $\pi^*$ for different choices of $\phi$: they are small, in $[1/8, 1/16]$. Hence, when $\varepsilon = 1/2$, a sufficient number of weight updates to ensure (23) and (37) is

$$\ell^*(T,c) = \frac{m}{8} \times \frac{1}{\vartheta^2\varrho^2} .$$

This happens to be a very reasonable constraint, given that the range of $\delta_j$ is of minute order compared to that in UNN, where $\delta_j$ can take on infinite values.

There is more: let $\ell_{\text{GNNB}}(T,c)$ and $\ell_{\text{UNN}}(T,c)$ denote the number of weight updates ensuring (37) (and thus ensuring (23) as well), respectively for GNNB and UNN. Inspecting Theorem 2.3 in [5] reveals that we have

$$\ell_{\text{GNNB}}(T,c) = \Theta\left(\frac{\ell_{\text{UNN}}(T,c)}{\varepsilon(1-\varepsilon)}\right) .$$

Hence, convergence rates of GNNB compete with those known for UNN.

Mathematical greedy algorithms [2] have a very wide scope, and they can be specialized to statistical learning with a high-level scheme which is close to the iterative scheme of boosting algorithms. Situating GNNB with respect to them is thus interesting and reveals quite a favorable picture, from the computational and convergence rate standpoints. These greedy algorithms are indeed computationally expensive, requiring at each iteration a local optimization of the classifier that GNNB does not require. Regarding convergence rates, the bound most relevant to our setting can be stated as follows, omitting unnecessary technical details and assumptions [2] (Theorem 3.1 and its proof): after $t$ iterations, the squared risk of the greedy output is no more than

$$\tau(t) = \beta\left(\frac{\kappa}{t} + \frac{t\ln(m)}{m}\right) ,$$

for some $\kappa, \beta$ that meet in general $\kappa \gg m$, and $\beta > 10^4$. This bound takes its minimum for some $t^*$ which is $\gg m$ in general. Even for this large $t^*$, the corresponding upperbound on the squared risk,

$$\tau(t^*) = 2\beta\sqrt{\kappa \times \frac{\ln(m)}{m}} ,$$

is significantly weaker than the guarantees of Theorem 4 and Corollary 1. Obviously however, our bounds rely on the **WLA**.

| category | name | $m$ | $C$ | ref. |
|---|---|---|---|---|
| small | Liver | 345 | 2 | [13] |
| | Ionosphere | 351 | 2 | [13] |
| | Pima | 768 | 2 | [13] |
| | Scene | 2407 | 6 | [25] |
| | Satellite | 6435 | 6 | [13] |
| | Segment | 2310 | 7 | [13] |
| | Cardio | 2126 | 10 | [13] |
| | OptDig | 5620 | 10 | [13] |
| | Letter | 2561 | 26 | [13] |
| large | Caltech | 30607 | 256 | [14] |
| | SUN | 108656 | 397 | [15] |

TABLE 2
Domains used in our experiments, ordered in increasing number of classes, and then examples.

## 6 Experiments

### 6.1 Domains

Experiments have been performed on a dozen domains summarized in Table 2. We have split the domains in small and large domains. Large domains have a significantly larger number of examples and classes. We refer the reader to the UCI machine learning repository for the related domains. We give a brief description of the "large" domains.

The Caltech [14] domain is a collection of 30607 images of 256 object classes. We adopt the Fisher vectors [26] encoding in order to describe these images as features vector . Fisher Vector are computed over densely extracted SIFT descriptors and local color features, both projected with PCA in a sub space of dimension 64. Fisher Vectors are extracted using a vocabulary of 16 Gaussian and normalized separately for both channels and then combined by concatenating the two features vectors. This approach leads to to a 4K dimensional features vector.

Finally, the SUN [5], [15] domain is a collection of 108656 images divided into 397 scenes categories. The number of images varies across categories, but there are at least 100 images per category. Each observation is represented as feature vector computed in the same way as for Caltech.

| top-1 accuracy (×100), Caltech | | | |
|---|---|---|---|
| | splits $n$ | | |
| iteration $t$ | 8 | 16 | 32 |
| 1 | 19.21 | 20.11 | 21.14 |
| 10 | 28.47 | 30.08 | 31.45 |
| 100 | 30.02 | 30.89 | 31.66 |
| 1000 | 30.38 | 31.52 | 32.65 |

| top-1 accuracy (×100), SUN | | | |
|---|---|---|---|
| | splits $n$ | | |
| iteration $t$ | 8 | 16 | 32 |
| 1 | 23.63 | 26.40 | 29.46 |
| 10 | 23.85 | 26.63 | 29.62 |
| 100 | 25.64 | 27.98 | 30.54 |
| 1000 | 25.64 | 27.97 | 30.56 |

| top-5 accuracy (×100), SUN | | | |
|---|---|---|---|
| | splits $n$ | | |
| iteration $t$ | 8 | 16 | 32 |
| 1 | 49.36 | 52.67 | 55.59 |
| 10 | 49.52 | 52.72 | 55.62 |
| 100 | 50.34 | 53.17 | 55.91 |
| 1000 | 50.33 | 53.19 | 55.92 |

TABLE 3
Performance of our divide-and-conquer approach on large domains for GNNB($\log$), using top-1 and top-5 accuracies.

Experiments are performed on a classical five-fold cross-validation basis, except for the large domains Caltech and SUN for which we have adopted the standardized approaches to use 30 random images from each class to train classifiers and the remaining for testing.

### 6.2 Metrics

We consider three types of metrics on small domains: the accuracy, which is one minus the Hamming risk (12, 13) and which is directly optimized by GNNB (Corollary 1), the recall and the F-measure, which is the harmonic average of precision and recall.

### 6.3 Algorithms

Because small domains do not raise the problem of execution time that large domains raise, we have chosen, on small domains, to perform an analysis as extensive as possible of the performances of GNNB. We have chosen twenty-two (22) algorithms that were tested on all nine domains, with each of the three metrics. The version of GNNB used is GNNB($\log$) (Row B in Table 1) with values of $k = 5, 10, 20, 50$. Contenders of GNNB can be put in five categories.

The first category consists of ordinary nearest neighbors, NN, tested with $k = 5, 10, 20, 50$, to assess what boosting in GNNB brings with respect to NN. Second, we consider UNN($\log$) with $k = 5, 10, 20, 50$, as another boosted NN contender, which performs, for this choice of BCL, approximations to the optimal boosted updates [5], that do not exist in closed form. The number $T$ of boosting iterations for GNNB and UNN is fixed to be the size of the training sample. Each example is boosted exactly once, and there is thus *no* inner optimization in the boosting rounds with respect to the weak learning assumption **WLA**.

For all nearest neighbors algorithms, we have made the choice to test a range of values for $k$ instead of optimizations of its value to allow fine grained observations on the behavior of these algorithms.

Third, we consider Stochastic Gradient Descent (SGD) [26], [27], [28], with four varying number of iterations. In the first, referred to as $\text{SGD}_1$, the number of iterations is equal to that of GNNB and UNN. In the second, $\text{SGD}_2$, number of iterations for SGD is fixed to be the "equivalent" to that of UNN and GNNB. Indeed, each iteration of SGD contributes to classify all examples in the training sample, while each iteration of UNN or GNNB contributes to classify $\theta(k)$ examples only. Thus, we need to make $\theta(m/k)$ iterations on UNN or GNNB for the classification of all examples to be eventually impacted. For this reason, if $T$ is the total number of boosting iterations in UNN and GNNB, then we perform $T \times k/m$ iterations of SGD. The two last runs of SGD, hereafter noted $\text{SGD}_3$ and $\text{SGD}_4$, consider a larger number of iterations, equal to two times the size of the training set in $\text{SGD}_3$ and three times the size of the training set in $\text{SGD}_4$. With those runs, we wanted to capture "limit" performances of stochastic gradient descent on small domains.

Fourth, we consider a popular boosting algorithm: ADABOOST [7], with four different flavors. In $\text{ADABOOST}_{\text{c2}}$, the weak learner is C4.5 [29] with depth-2 threes; in $\text{ADABOOST}_{\text{c3}}$, the weak learner is C4.5 with depth-3 threes. Notice that, in these two cases, the inner boosting rounds are optimized to fit a classifier that minimizes the expected criterion at hand (in C4.5, it is $-\phi$ in row B of Table 1), which is, as discussed above, not the case of GNNB. To have something comparable from the algorithmic standpoint in GNNB, we should have *e.g.* chosen to leverage at each iteration the example that maximizes $\varrho$ and/or $\vartheta$ in the **WLA**. For this reason, we have also tested ADABOOST with a non-optimized weak learner, which returns random trees. In $\text{ADABOOST}_{\text{r3}}$, these trees have depth 3, and in $\text{ADABOOST}_{\text{ru}}$, these trees have unbounded depth. In all four flavors of ADABOOST, the number of boosting rounds equals that of GNNB and UNN.

Fifth and last, we have considered two flavors of support vector machines, the first of which is affordable on small domains (but out of reach on our largest domains), non-linear SVM with radial basis function kernel [30] in which the regularization parameter and the bandwidth are further optimized by a five-fold cross-validation on the training sample. We refer to them as $\text{SVM}_{\text{RBF}}$. The second flavor is linear SVM, $\text{SVM}_{\text{L}}$, which remains the main SVM algorithm which can be run on very large datasets.

On large domains, we have tested GNNB against the contenders that scored top in the small domains or were easily scalable to large domains: NN, UNN, SGD.

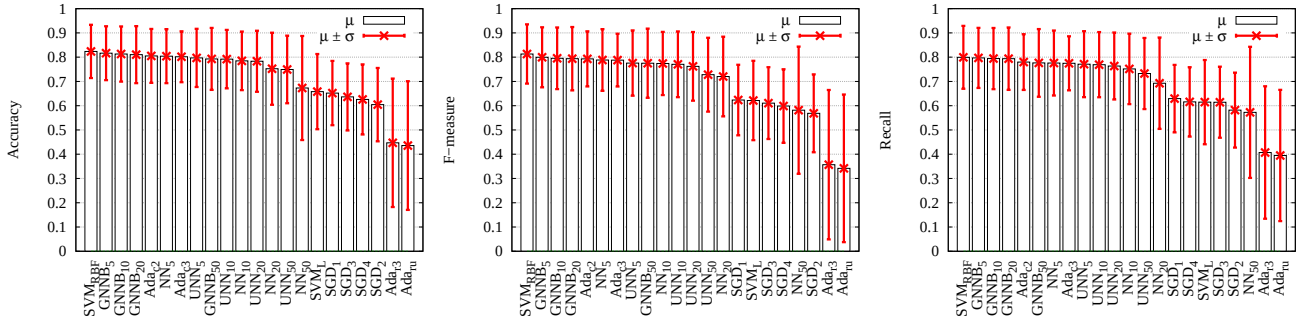


Fig. 3. Average ($\mu$) $\pm$ Standard deviation ($\sigma$, in red) for the accuracy (left), F-measure (center) and recall (right) over the 9 small domains for the 21 algorithms. In each plot, algorithms are ordered from left to right in decreasing average of the metric at hand.

We have also tried $\text{SVM}_{\text{LLC}}$, that is, linear SVM with locality-constrained linear coding LLC [31]. LLC is a coding scheme which puts emphasis on locality constraints for representing data, and brings desirable byproducts such as sparsity.

### 6.4 A divide-and-conquer optimization of GNNB

It is well known that NN classifiers suffer of the curse of dimensionality [32], hubs [33], so that the accuracy can decrease when increasing the size of descriptors. This may also affect GNNB, in particular on large domains like SUN and Caltech. Fisher vectors employ powerful descriptors but they generate a space with about 4K dimension for 32 gaussians, which could impair GNNB performance. Our approach relies on a property of classification-calibrated losses that one can get simple posteriors estimators from the classifier's output, based on the matching posterior $\hat{p}_{\phi,h}$ in (5) (see [10], [5], and the right plot Figure 1) . The method we propose consists in (i) splitting the set of descriptors, (ii) compute posteriors over each of these sets, and finally (iii) average the posteriors over all splits. The set of Fisher descriptors is split in a regular set of $n \in \{8, 16, 32\}$ sub-descriptors; each set is normalized in $L_1$ or $L_2$ norm. Finally, posteriors are combined linearly, with an arithmetic average.

Table 3 presents the results obtained on our large domains. Results in Table 3 show that increasing $n$, the number of splits, always improves the performances of GNNB, in a range between $1\%$ and $6\%$, the largest improvements being obtained for the largest domain (SUN). We have also checked that increasing the number of iterations still keeps this pattern, which is thus robust to both variations in $n$ and the total number of boosting iterations $t$. We have witnessed in some cases differences that become much more important with the increase in $t$. For example, after 7650 iterations on Caltech, GNNB's top-1 accuracy becomes respectively $31.91\%$, $33.79\%$ or $36.13\%$ for $n = 8, 16$ and $n = 32$.

In the results of the following subsections, GNNB is ran with $n = 32$ splits. To remain fair with UNN, we have also carried out the same $n = 32$ splitting strategy, after having checked that it improves the performances of UNN as well.

### 6.5 Results on small domains

#### 6.5.1 Results on average metrics

Figure 3 presents the average results obtained for the 22 algorithms on the 3 metrics: accuracy, F-measure and recall. Over all three metrics, one can notice that the algorithms cluster in three groups:

- the group of the best performing algorithms, with non-linear and mostly optimized large margin algorithms: $\text{SVM}_{\text{RBF}}$, GNNB (all $k$s), UNN (all $k$s), ADABOOST+C4.5, and NN with $k = 5, 10, 20$;
- a group of algorithms that perform not as well as the first, with mostly linear classification algorithms: $\text{SVM}_{\text{L}}$, all SGD algorithms and NN with $k = 50$;
- the group of algorithms that perform the worst of all, containing randomized large margin classification: ADABOOST with random trees.

Several observations come to mind. First, the performances of all nearest neighbor methods (GNNB, UNN, NN) decrease with $k$, in the range of values selected. Second, boosting the nearest neighbors (GNNB, UNN) dampens the degradation of performances. Third, GNNB performs the best of all three kinds of nearest neighbor methods, from the standpoint of all three metrics.

In fact, GNNB performs on par with $\text{SVM}_{\text{RBF}}$, for a wide range of $k$ $(5, 10, 20)$. The comparison with $\text{ADABOOST}_{\text{r3}}$ and $\text{ADABOOST}_{\text{ru}}$ is clear and final, as regardless of $k$ and for all metric, GNNB is better by more than .2 points on average; finally, GNNB performs also slightly better than ADABOOST + C4.5 (for $k = 5, 10, 20$). Since $\text{SVM}_{\text{RBF}}$, ADABOOST+trees and GNNB can all be viewed as large margin non-linear methods, the fact that we do not optimize GNNB from the standpoint of $k$, $\vartheta$ or $\varrho$ (in the **WLA**) makes it the method, out of the three, which is the easiest to scale, and further optimize, to large domains.

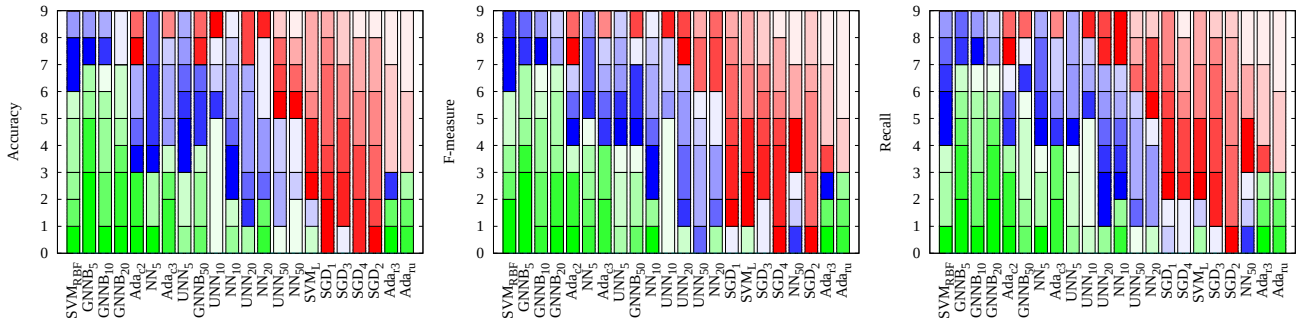


Fig. 4. Ranking results: colors indicate the number of times an algorithm ranked among the top-tier (green), second-tier (blue) and third-tier (red, depicting the worst 8 algorithms) among all 22 algorithms, over the 9 small domains. For each color, the lighter the tone, the higher (and worse) the rank. For example, dark green refers to to rank 1, the lightest green to rank 7 and dark blue to rank 8. In each plot, algorithms are ordered from left to right in decreasing average of the metric at hand.

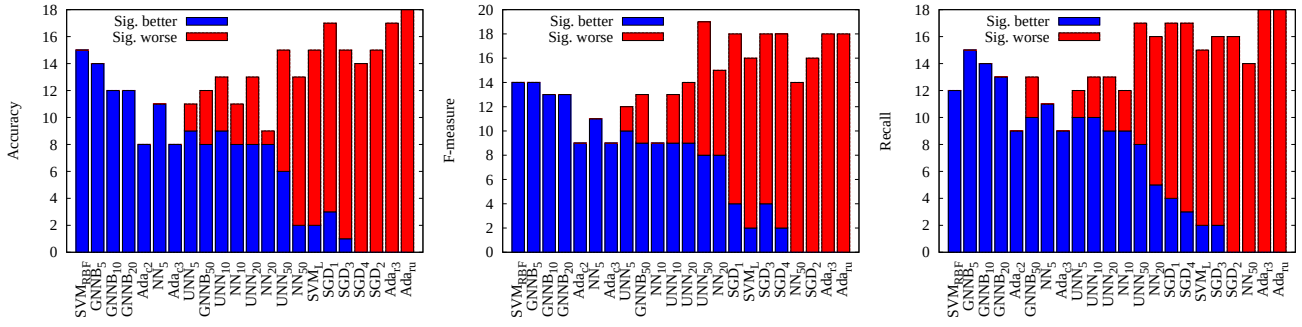


Fig. 5. Ranking results contd: number of times each algorithm performed significantly better than the others (blue) or worse (red) according to a Student paired $t$-test ($p = .1$). In each plot, algorithms order follow Figures 3 and 4 (see text).

#### 6.5.2 *Ranking results*

To drill down into these general results, we have also computed the global ranking results of each algorithm, recording the number of times each ranked first, second, third and so on, on the 9 domains. Figure 4 provides the global ranking results for all 22 algorithms and all 3 metrics. These plots allow to nuance the results on average metrics, with the following observations.

First, there is a subgroup in the group of the best performing algorithms according to the average metrics, which is the best according to ranking: $\text{SVM}_{\text{RBF}}$ and GNNB ($k = 5, 10, 20$). In this group, it appears that GNNB tends to be ranked higher than $\text{SVM}_{\text{RBF}}$, for a wide range of $k$ $(5, 10, 20)$, and this is particularly visible for F-measure and recall. From the recall standpoint, GNNB is almost always in top-tier results, while $\text{SVM}_{\text{RBF}}$ is more often in the second-tier.

Second, SGD performs poorly from the ranking standpoint, as all flavors mostly score among the third-tier results. We also observe that SGD performances are not monotonous with the number of iterations, as $\text{SGD}_1$ performs the best of all, both from the average and ranking standpoints. Linear classification methods tend to perform poorly, as displayed by $\text{SVM}_{\text{L}}$'s ranking results, very similar to those of stochastic gradient descent. If we compare ranking results with those of ADABOOST+random trees, which is the worst of all algorithms from the expected metrics standpoint, then the ranking results display that SGD is in fact more often in the third-tier of all 22 algorithms.

Finally, ADABOOST with random trees sometimes scores very well among algorithms, and its ranking patterns, along with average metrics' results, clearly display that the poor average results are essentially due to some domains for which replacing the randomized weak learner by an optimized one would make a big difference, as a random trees weak learner yields the worst performances of all 22 algorithms, while C4.5 brings at least second-tier performances.

To bring statistical validation to these ranking results, we performed Student paired $t$-test comparison for each algorithm against all others ($22 \times 21 = 462$ comparisons), recording those for which we can reject the null hypothesis (that the per-domain difference has zero expectation) for level $p = .1$, and then clustering the "significant" differences as to whether they are in favor, or in disfavor, of the algorithm at hand. Figure 5 summarizes the results obtained, for all three metrics. They allow to cluster algorithms in three: those that are never significantly outperformed

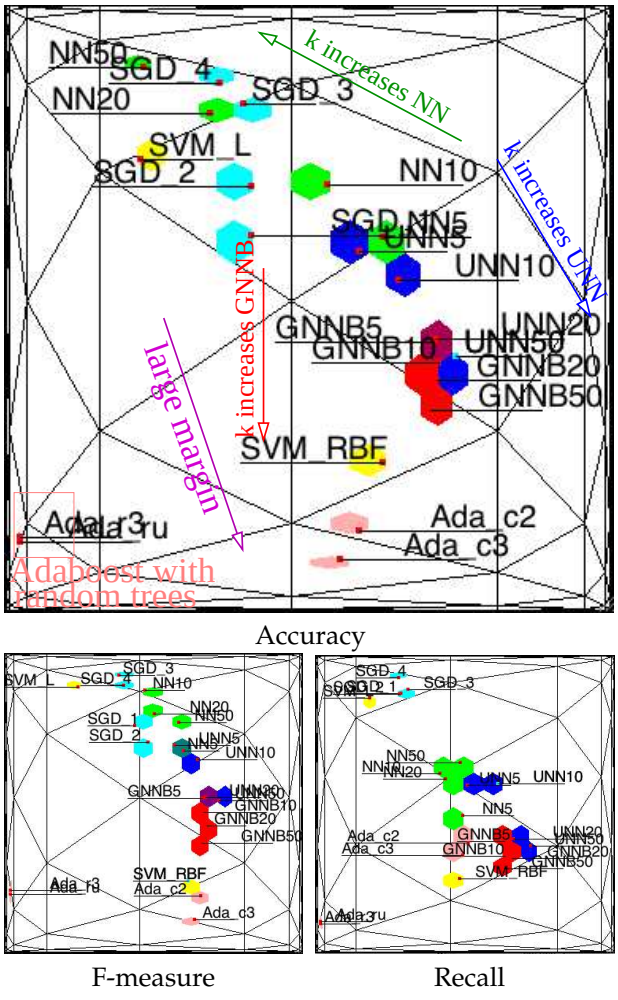


Fig. 6. Manifold classification patterns for the accuracy (up, commented), F-measure (bottom left) and Recall (bottom right), for all 22 algorithms (see text); colors in hexagons cluster types of algorithms: red = GNNB, yellow = SVM, pink = ADABOOST, cyan = SGD, blue = UNN, green = NN (see text and [34] for details).

(GNNB for $k = 5, 10, 20$, $\text{SVM}_{\text{RBF}}$, ADABOOST+C4.5, NN for $k = 5$), those that never significantly outperform ($\text{SGD}_2$, NN for $k = 50$, ADABOOST+random trees), and the rest of the algorithms. They confirm that GNNB performs on par with or better than optimized large margin non-linear algorithms ($\text{SVM}_{\text{RBF}}$, ADABOOST+C4.5), and this pattern holds for a wide range of values of $k$.

### 6.5.3 Classification patterns

The algorithms we have tested on small domains are representative of major families of supervised classification algorithms, ranging from linear to non-linear, induced to non-induced, including large margin classification methods, stochastic algorithms, and so on. To get a *qualitative* picture of the performances of GNNB, we have learned a manifold on the algorithms' results, one for each of the three metrics, in the following way.

To get rid of the quantitative differences, we have normalized results to zero mean and unit standard deviation in each domain. Then, a manifold was learned using a standard procedure, using the normalized cosine similarity measure [35] to get a symmetric similarity matrix, and computing the second and third leading eigenvector of the Markov chain from that similarity matrix [36], [35]. We checked that the corresponding eigenvalues were significantly larger than the following ones.

The corresponding manifolds are displayed in Figure 6, using an information-geometric focus+context display [34] in which the focus area is the center of the square. To help the unfamiliar reader capture the distortions of the display, plots also display in the background the mapping of a regular equilateral triangular tiling of the plane. The focus on data has been chosen so as to zoom on the part of the manifold with the largest number of algorithms.

The main observation from the plots, which cannot be observed in the average metrics and ranking experiments, is that the recall plot is much different from the accuracy and F-measure plots, thereby displaying that precision and recall have very different patterns among algorithms. The recall plot clusters the algorithms in three categories: linear classification (top-left, SGD, $\text{SVM}_{\text{L}}$), randomized boosting (ADABOOST+random trees, bottom left), and the rest of the algorithms (center). The accuracy and F-measure plot make a clear distinction between non-linear large margin "optimized" (down-right), non-linear large margin "random" (down-left) and linear (up). Looking at nearest neighbor algorithms as $k$ increases reveals that boosted nearest neighbor algorithms (UNN, GNNB) tend to behave more and more like large margin classification algorithms as $k$ increases, while vanilla NN tends to behave more and more like linear classification algorithms as $k$ increases. This observation for NN is consistent with the simple example that sampling two spherical Gaussians with identical variance (one for each class) makes a non-linear frontier for $k, m \ll +\infty$, which tends to a linear one as both parameters tend to $+\infty$.

### 6.5.4 Training times

We have computed the training times for GNNB (all $k$s), $\text{SVM}_{\text{RBF}}$ and ADABOOST+C4.5 (depth-3 trees), that represent the top-5 or top-6 algorithms in terms of average metric performances. To summarize these results, we have computed the ratio between training times for each domain and each value of $k$, for $\text{SVM}_{\text{RBF}}$ to GNNB, and ADABOOST+C4.5 to GNNB. As already displayed for UNN [5], the ratios are clearly in favor of GNNB. We obtained a synthetic and accurate picture of these advantages by regressing the ratio against $1/k$, that is, computing the regression coefficients $a, b$ for $\rho = (a/k) + b$, where $\rho$ is *e.g.* the ratio for the $\text{SVM}_{\text{RBF}}$ training time to GNNB training time,

averaged over all domains, and then computed for each $k$. The results, that we give with the coefficient of determination $r^2$, are (t.t. = training time):

$$\frac{\text{t.t.(SVM}_{\text{RBF}})}{\text{t.t.(GNNB)}} \approx \frac{851}{k} + 49 \quad (r^2 = 0.96) \ ,$$
$$\frac{\text{t.t.(ADABOOST+C4.5)}}{\text{t.t.(GNNB)}} \approx \frac{9547}{k} + 398 \quad (r^2 = 0.97) \ .$$

These regressions mean that, regardless of the value of $k$, $\text{SVM}_{\text{RBF}}$'s training time is at least roughly 50 times that of GNNB, while ADABOOST+C4.5's training time is at least roughly 400 times that of GNNB. On testing, the variation with $k$ is clearly dampened, but $\text{SVM}_{\text{RBF}}$'s testing time is still at least 300 times that of GNNB, while ADABOOST+C4.5's is at least 2000 times that of GNNB. These ratios are in good agreement with those observed in favor of UNN against $\text{SVM}_{\text{RBF}}$ and ADABOOST+stumps [5].

#### 6.5.5 Summary for small domains

The results obtained on the 9 small domains and with the 3 metrics, for the 22 algorithms, as observed from the qualitative and quantitative (averages, ranks, times) standpoints, bring the following general observations. First, GNNB scores among the top algorithms and performs on par with, or better than, optimized machineries like non-linear SVM or ADABOOST+trees, and it beats these latter approaches, from the training/testing times standpoint, by factors that range from tens to thousands of times. These excellent performances go hand in hand with the desirable property that results are remarkably stable against reasonable variations of $k$, which appears to be, in comparison, clearly not the case for UNN.

### 6.6 Results on large domains

We have used the instantiation of SGD that performed the best on small domains, $\text{SGD}_1$, and the number of iterations of GNNB and SGD is 6000. For each large domain, we split the analysis between the comparison of GNNB vs UNN, and GNNB vs the rest of the algorithms.

#### 6.6.1 Results on Caltech

The two left plots of Figure 7 display the results of GNNB vs UNN on Caltech. We have chosen to put emphasis on the relative variations of GNNB wrt UNN, to get an immediate picture of which algorithm performs the best and by what amount.

The plots of Figure 7 display that GNNB outperforms UNN, and this phenomenon is dampened as $k$ increases. This is quite in accordance with the universal consistency of the algorithms (Lemma 4). For $k = 100$, the improvement of GNNB from the accuracy and recall standpoint exceed $+20\%$, and it is reduced to $+10\%$ for $k = 200$.

| | NN | GNNB | $\text{SGD}_1$ | $\text{SVM}_{\text{LLC}}$ | | |
|---|---|---|---|---|---|---|
| $f$ | 4K | 4K | 4K | 4K | 4×4K | 5×4K |
| Acc. | 25.50 | 36.40 | 36.00 | 27.99 | 35.18 | 36.76 |
| F-m. | 20.97 | 29.24 | 30.87 | 24.00 | 31.67 | 33.33 |
| Rec. | 17.13 | 31.47 | 31.35 | 22.00 | 28.66 | 30.41 |

TABLE 4
Results on Caltech (accuracy, F-measure and recall are $\times 100$). $f$ is the number of features, and $k = 200$ for NN, GNNB.

| | NN | GNNB | $\text{SGD}_1$ |
|---|---|---|---|
| Acc. | 20.92 | 30.16 | 32.20 |
| F-m. | 17.39 | 27.02 | 30.96 |
| Rec. | 23.39 | 34.32 | 35.53 |

TABLE 5
Results on SUN according (accuracy, F-measure and recall are $\times 100$). $k = 200$ for NN, GNNB.

Table 4 compares GNNB to NN, $\text{SGD}_1$ and LLC encoding for linear SVM using the same codebook as [31]. LLC produces a very large number of descriptors compared to the 4K Fisher vectors used in the other approaches, and a significant part of the improvement is in fact due to this very large description space [37]. In order to make fair comparisons with the other techniques that rely on the 4K descriptors, we have extracted the two first layers of descriptors of LLC, of size 4K and 4×4K, to analyze $\text{SVM}_{\text{LLC}}$ over 4K descriptors, 4×4K descriptors and 4K+4×4K = 5K descriptors.

The accuracy results show that GNNB tops NN and $\text{SGD}_1$, and beats $\text{SVM}_{\text{LLC}}$ until 16K descriptors. It is only when $\text{SVM}_{\text{LLC}}$ uses five times the number of descriptors of GNNB that it beats GNNB. In fact, when using the same description size as the other algorithms, LLC encoding is beaten from the standpoint of all metrics by GNNB and $\text{SGD}_1$. $\text{SGD}_1$ performs well from the standpoint of the F-measure, and performs on par with GNNB from the recall standpoint.

#### 6.6.2 Results on SUN

The comparison between GNNB and UNN (Figure 7, right plots) displays the same patterns as for Caltech: as $k$ increases, the improvements of GNNB wrt UNN are dampened, yet they stay this time always in favor of GNNB, and the improvements are more pronounced: they range in between $10\%$ and $30\%$ for $k = 100$, for each metric. We conjecture that this comes from the fact that SUN is significantly larger than Caltech: to reduce further the improvement of GNNB with respect to UNN, one should probably test much larger $k$s — a tricky and time-consuming task, as the variations in improvements are not monotonous with $k$ —, at the obvious expense of larger training times and storage space (or memory) requirements.

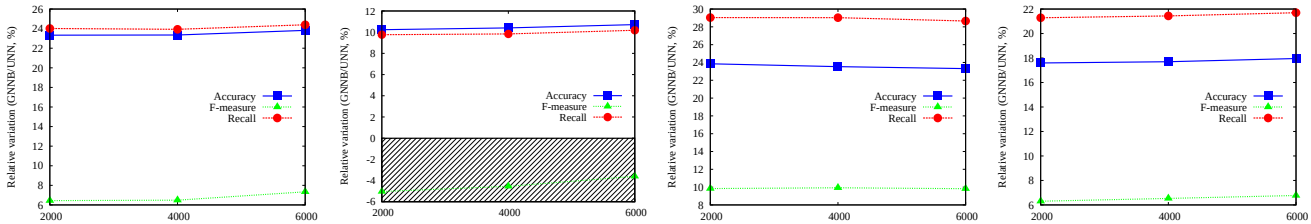


Fig. 7. Relative variation (in %) of GNNB vs UNN, for large domains and for each metric, expressed as a function of the number of boosting rounds $t$. Positive values indicate better results for GNNB; a dashed rectangle indicates the zone of negative values.

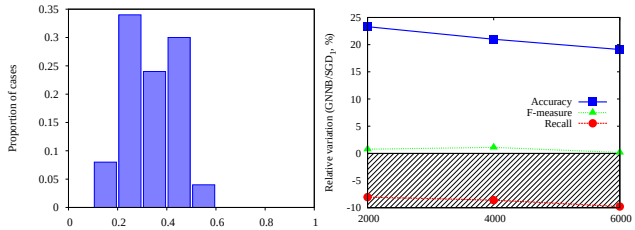


Fig. 8. Left: frequency of cases among classes for the proportion of examples used per class by GNNB; Right: GNNB$^*$ ($k = 200$) vs SGD$_1$ (conventions follow Fig. 7).

Table 5 compares the performances of GNNB vs NN and SGD$_1$. This time, SGD$_1$ beats GNNB from the standpoint of all metrics. This observation has to be taken with a pinch of salt, as the experimental setting for large domains disfavors GNNB. Indeed, GNNB, like NN, is a local classifier, and for such kinds of methods, the random sampling setting adopted is a random prototype selection method which filters out more than 80% of the dataset, increasing significantly the distances between nearest neighbors and impairing the estimators quality. Furthermore, GNNB as used so far does not carry out any optimization on the learning examples: it iterates through each of them, leveraging each of them exactly once. On the other hand, the effects of subsampling can be quite minor on linear separators: for example, provided a linear separator exists with minimal margin $\gamma$, sampling $\tilde{\Omega}(\gamma^{-2})$ examples (the tilde notation hides dependences in other parameters) at random still guarantees with high probability the existence of a linear separator with $\Omega(\gamma)$ margin and small true risk [38].

To get a more reliable picture of the performances of GNNB on our largest domain, we thus have considered a naive optimization of the weak index chooser WIC in GNNB, and tested it in an experimental setting computationally affordable for GNNB and less in disfavor than the former one. The new WIC in GNNB returns the index of the example with the largest current $|\delta_j|$. This version of GNNB was denoted GNNB$^*$ in Subsection 4.4. To alleviate the discrepancies due to the experimental setting, we have tested GNNB$^*$ on a 50% holdout of the SUN database, recording for each class the percentage of examples actually used in training, *i.e.* leveraged or reweigthted. Then, we run SGD$_1$ using the same former experimental setting (since the new one for GNNB was computationally too heavy), replacing however the number of examples used in the former setting (30) by the number of examples corresponding to the average frequency of the whole domain used by GNNB$^*$ (capped by the class size for the smallest ones), to ensure that the data available to SGD was not smaller than for GNNB. The left plot in Figure 8 provides the histogram of the proportion of data used to learn each class. The expectation in $x$ is roughly 33%, and thus we ran SGD$_1$ using at each iteration roughly 46 examples per class, which is the average of 33% of the 50% of each class. The right plot in Figure 8 summarizes the improvements of GNNB$^*$ with respect to SGD$_1$. One sees this time that even when the recall of GNNB$^*$ is smaller than that of SGD$_1$, the accuracy is now comparatively significantly better. While optimizing WIC in GNNB, or GNNB itself, was not the purpose of this paper, this simple experiment displays that there may be significant room for further improvement of GNNB, in particular for large scale learning. This is interesting as GNNB belongs to the small set of algorithms that would require comparatively little computational tuning to be affordable on domains even larger.

## 7 Conclusion

We proposed in this paper a simple Newton-Raphson leveraging scheme for nearest neighbors to optimize any even, twice differentiable proper scoring rule. This scheme has guaranteed convergence rates under the boosting framework that compete with those known for non-gentle approaches like UNN [6]. To the best of our knowledge, those convergence rates in the boosting framework are knew for Gentle Newton-Raphson boosting approaches [3]. We also show that our algorithm, GNNB, complies with weak universal consistency. Experiments tend to display that GNNB significantly outperforms UNN by converging faster to better solutions. Experiments on small domains display that GNNBperforms on par with or better than powerful non-linear large margin

approaches like non-linear SVM and Adaboost+C4.5. Experiments on large domains, on which these powerful approaches are ruled out because of their computational costs, display that GNNB provides a simple and competitive alternative to stochastic gradient algorithms. We have shown that real-valued classification in GNNB goes hand in hand with a universally consistent posteriors estimation throughout balanced convex losses, and we have shown how to exploit this link experimentally for a divide-and-conquer scheme which demonstrates superior performances over vanilla GNNB. This scheme averages the posteriors independently of the choice of the permissible function, and we believe that further improvements of the scheme could be obtained by tuning the average. Finally, we have tested manifold learning approaches to assess global qualitative comparisons of large numbers of algorithms from observed performances. As learning algorithms are rapidly becoming more numerous and complex, we believe that such techniques may be interesting for large-scale comparisons, and help the design of new algorithms.

## 8 Acknowledgments

The authors would like to thank the reviewers for insightful comments that helped to improve this work. The code used (GNNB and manifold learning) are available upon request to M. Barlaud and R. Nock.

## References

[1] L. Breiman, J. H. Freidman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, Wadsworth, 1984.

[2] A.-R. Barron, A. Cohen, W. Dahmen, and R.-A. DeVore, "Approximation and learning by greedy algorithms," *Ann. of Stat.*, vol. 26, pp. 64–94, 2008.

[3] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression : a Statistical View of Boosting," *Ann. of Stat.*, vol. 28, pp. 337–374, 2000.

[4] L. Cucala, J.-M. Marin, C.-P. Robert, and D.-M. Titterington, "A Bayesian reassessment of nearest-neighbor classification," *J. of the Am. Stat. Association*, vol. 104, pp. 263–273, 2009.

[5] R. Nock, P. Piro, F. Nielsen, W. Bel Haj Ali, and M. Barlaud, "Boosting $k$-NN for categorization of natural scenes," *International J. of Computer Vision*, vol. 100, pp. 294–314, 2012.

[6] P. Piro, R. Nock, F. Nielsen, and M. Barlaud, "Leveraging $k$-NN for generic classification boosting," *Neurocomputing*, vol. 80, pp. 3–9, 2012.

[7] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learning*, vol. 37, pp. 297–336, 1999.

[8] N. García-Pedrajas and D. Ortiz-Boyer, "Boosting $k$-nearest neighbor classifier by means of input space projection," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10570–10582, 2009.

[9] P. Bartlett, M. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *J. of the Am. Stat. Association*, vol. 101, pp. 138–156, 2006.

[10] R. D'Ambrosio, R. Nock, W. Bel Haj Ali, F. Nielsen, and M. Barlaud, "Boosting nearest neighbors for the efficient estimation of posteriors," in *Proc. of the 23 $^{rd}$ ECML-PKDD*, 2012, pp. 314–329.

[11] R. Nock and F. Nielsen, "Bregman divergences and surrogates for learning," *IEEE PAMI*, vol. 31, no. 11, pp. 2048–2059, 2009.

[12] T. Gneiting and A. Raftery, "Strictly proper scoring rules, prediction, and estimation," *J. of the Am. Stat. Association*, vol. 102, pp. 359–378, 2007.

[13] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

[14] Gregory Griffin, Alex Holub, and Pietro Perona, "Caltech-256 object category dataset," 2007.

[15] J. Xiao, J. Hays, K.-A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Proc. of IEEE CVPR*, 2010, pp. 3485–3492.

[16] R. Nock and F. Nielsen, "On the efficient minimization of classification-calibrated surrogates," in *NIPS*21*, 2008, pp. 1201–1208.

[17] E. Vernet, R.-C. Williamson, and M.-D. Reid, "Composite multiclass losses," in *NIPS*24*, 2011, pp. 1224–1232.

[18] P. Grünwald and P. Dawid, "Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory," *Ann. of Stat.*, vol. 32, pp. 1367–1433, 2004.

[19] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin : a new explanation for the effectiveness of voting methods," *Ann. of Stat.*, vol. 26, pp. 1651–1686, 1998.

[20] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer, 1996.

[21] M. Sebban, R. Nock, and S. Lallich, "Boosting Neighborhood-Based Classifiers," in *Proc. of the 18 $^{th}$ ICML*, 2001, pp. 505–512.

[22] M. Sebban, R. Nock, and S. Lallich, "Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problems," *JMLR*, vol. 3, pp. 863–885, 2003.

[23] S. Kakade, S. Shalev-Shwartz, and A. Tewari, "Applications of strong convexity–strong smoothness duality to learning with matrices," Tech. Rep. CoRR abs/0910.0610, CoRR, 2009.

[24] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *JMLR*, vol. 6, pp. 1705–1749, 2005.

[25] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown, "Learning multi-label scene classification," *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[26] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, "Towards good practice in large-scale learning for image classification," in *Proc. of IEEE CVPR*, 2012 (to appear).

[27] Léon Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. of COMPSTAT*, pp. 177–186. Springer, 2010.

[28] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro, "Pegasos: Primal estimated sub-gradient solver for svm," in *ICML'07*. ACM, 2007, pp. 807–814.

[29] J. R. Quinlan, *C4.5 : programs for machine learning*, Morgan Kaufmann, 1993.

[30] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[31] J. Wang, J. Yand, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. of IEEE CVPR*, 2010, pp. 3360–3367.

[32] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft, "When is nearest neighbor meaningful?," *Proc. of the 7$^{th}$ ICDT*, pp. 217–235, 1999.

[33] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović, "Hubs in space: Popular nearest neighbors in high-dimensional data," *JMLR*, vol. 9999, pp. 2487–2531, 2010.

[34] R. Nock and F. Nielsen, "Information-Geometric Lenses for multiple Foci+Contexts interfaces," in *6 $^{th}$ SIGGRAPH Asia — Technical Briefs*, 2013, p. accepted.

[35] R. Nock, F. Nielsen, and E. Briys, "Non-linear book manifolds: learning from associations the dynamic geometry of digital libraries," in *Proc. of the 13$^{th}$ ACM/IEEE JCDL*, 2013, pp. 313–322.

[36] M. Meilă and J. Shi, "Learning segmentation by random walks," in *NIPS*14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. 2001, MIT Press.

[37] K. Chatfield, V.-S. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *Proc. of the 20 $^{th}$ British Machine Vision Conference*, 2011, pp. 1–12.

[38] A. Blum, "Random projection, margins, kernels, and feature-selection," in *Subspace, Latent Structure and Feature Selection*, Craig Saunders, Marko Grobelnik, Steve Gunn, and John Shawe-Taylor, Eds., vol. 3940 of *Lecture Notes in Computer Science*, pp. 52–68. Springer Verlag, 2006.